



Application Note

Library for file transfer with FTP for XControl PLCs

08/05 AN2700K34GB V1.0

© Moeller GmbH, Bonn

Author: Matthias Blümling

All brand and product names are trademarks or registered trademarks of the owner concerned.

All rights reserved, including those of the translation.

No part of these application notes may be reproduced in any form (printed, photocopy, microfilm or any other process) or processed, duplicated or distributed by means of electronic systems without the written permission of Moeller GmbH, Bonn.

Status: April 2001

Subject to alteration

Table of Contents

1	GENERAL	3
1.1	FUNCTION	3
1.2	FIELD OF APPLICATION	3
1.3	HARDWARE REQUIREMENTS	4
1.4	SOFTWARE REQUIREMENTS	4
2	THE FTP LIBRARY	5
2.1	INCORPORATING THE FTP LIBRARY	5
2.2	FUNCTION BLOCKS AND PARAMETERS	5
2.2.1	<i>FTP_Connect</i>	6
2.2.2	<i>FTP_Username</i>	6
2.2.3	<i>FTP_Password</i>	7
2.2.4	<i>FTP_Disconnect</i>	7
2.2.5	<i>FTP_ChangeDirectory</i>	8
2.2.6	<i>FTP_WorkingDirectory</i>	8
2.2.7	<i>FTP_MakeDirectory</i>	9
2.2.8	<i>FTP_DeleteDirectory</i>	9
2.2.9	<i>FTP_DeleteFile</i>	10
2.2.10	<i>FTP_UploadFile</i>	10
2.2.11	<i>FTP_DownloadFile</i>	11
2.2.12	<i>FTP_GetLastError</i>	11
2.2.13	<i>FTP_GetLastServerResponse</i>	12
2.2.14	<i>Global parameters</i>	12
3	COMMISSIONING	13
3.1	INCLUSION OF FTP INTERFACING UNDER THE X SERIES PLCs	13
3.2	USING THE FTP LIBRARY	13
3.2.1	<i>Logging on to the server</i>	13
3.2.2	<i>Directory functions</i>	13
3.2.3	<i>File functions</i>	14
3.2.4	<i>Logging off FTP the server</i>	14
3.2.5	<i>Function FTP_GetLastError</i>	14
3.2.6	<i>Function FTP_GetLastServerResponse</i>	16
4	VERSION INDEX	17

1 General

1.1 Function

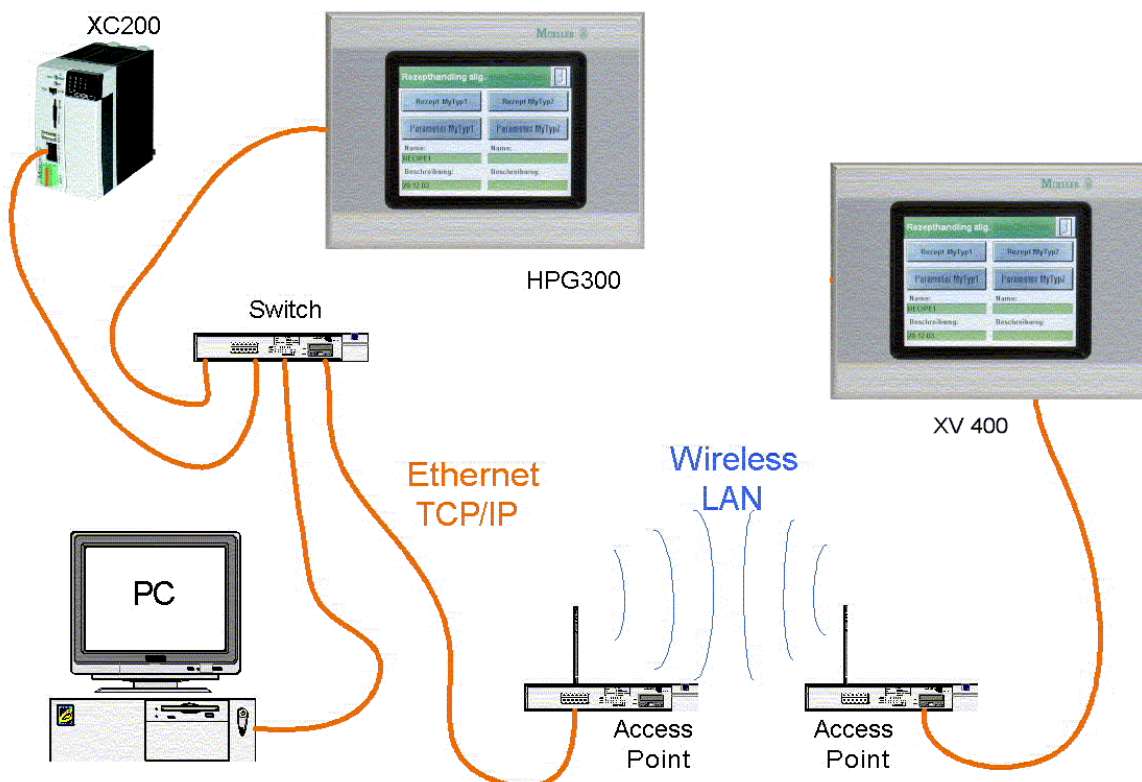
FTP client, for use with XControl PLCs and MC-HPG.

- Logging FTP servers on and off
- Uploading and downloading files
- Deleting files
- Scanning the current folder
- Creating and deleting folders

1.2 Field of application

Application-controlled file transfer between PLCs (also to or from the PC).

Example: Uploading and downloading readings files, recipes, configurations, etc.



1.3 Hardware requirements

XControl PLC	from OS version
MC-HPG 200	V1.34
MC-HPG 300	V1.34
XC 200	V1.03.02
XV 400	V2.04
XVC 600	V1.34
XCC 600	V1.34

1.4 Software requirements

You will need XSoft from version 2.3.3.

Required libraries:

- Standard.lib
- SysLibCallback.lib
- SysLibSockets.lib from version 2.4.0.6
- SysLibFile.lib

2 The FTP library

2.1 Incorporating the FTP library

Use the library manager to add library FTP_Client.lib to the application, from where they can then be called. You must add the additionally required libraries Standard.lib, SysLibSockets.lib, SysLibCallback.lib and SysLibFile.lib, if they aren't already added.

2.2 Function blocks and parameters

Note: The internal function can be called only through the functions in the FTP library, not directly. They can not be called directly by the user program.

Note: Only one FTP function can be busy per cycle. Before calling a further FTP function, the previous function must have been fully processed.
If the function has output variable *xBusy*, wait until this variable is FALSE before starting the next FTP function.

All function blocks are started with a positive edge at Boolean input variable *xExecute*.

Output variable *xBusy* is TRUE as long as the function block is busy. During this time, no further FTP commands must be run.

Output variables *xError* and *xDone* show the result of the function call for one cycle when *xExecute* has reverted to FALSE again. If *xExecute* is still TRUE, *xDone* and *xError* are reset to FALSE with the falling edge of *xExecute*.

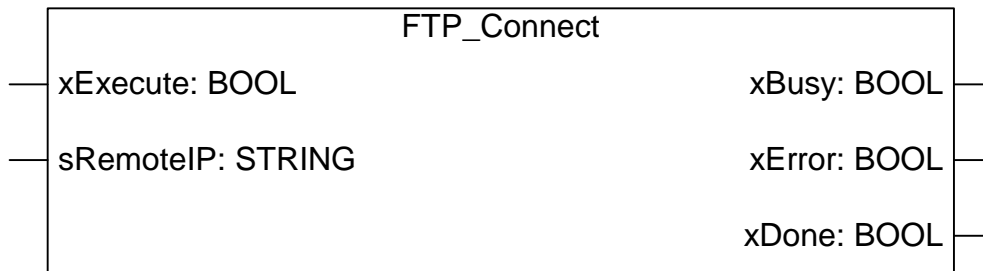
The busy period depends on the function block and on the set task interval and the size of the transferred file.

Theoretically, the size of the transferred file is limited only by the available memory in the PLC, MMC or USB stick.

Output variable *xError* indicates any errors. Together with *xError*, FTP_Client_wErrorID outputs a global error ID. The 2-digit code provides the error cause to simplify troubleshooting (see also 3.2.5)

2.2.1 FTP_Connect

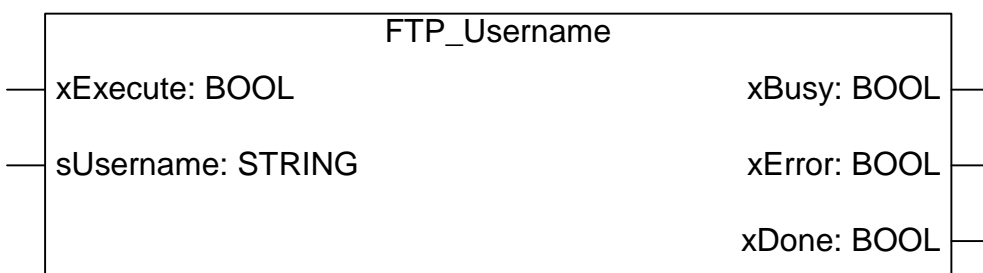
This function is used to set up a connection between the PLC and an FTP server.



<i>Var_Input</i>	
xExecute	Function is started with a positive edge
sRemoteIP	IP of FTP server in standard dotted format
<i>Var_Output</i>	
xBusy	Indicates that the function is busy
xDone	Indicates successful execution of function
xError	Indicates an error

2.2.2 FTP Username

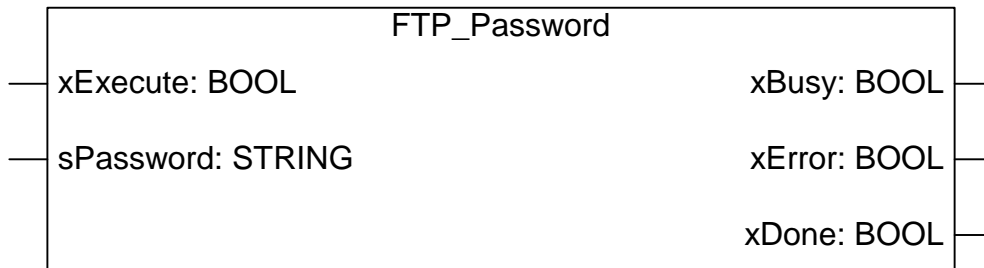
This function transmits a user name to the server. It is required for logging in.



<i>Var_Input</i>	
XExecute	Function is started with a positive edge
sUsername	Username. Up to 80 characters
<i>Var_Output</i>	
xBusy	Indicates that the function is busy
xError	Indicates an error
xDone	Indicates successful execution of function

2.2.3 FTP_Password

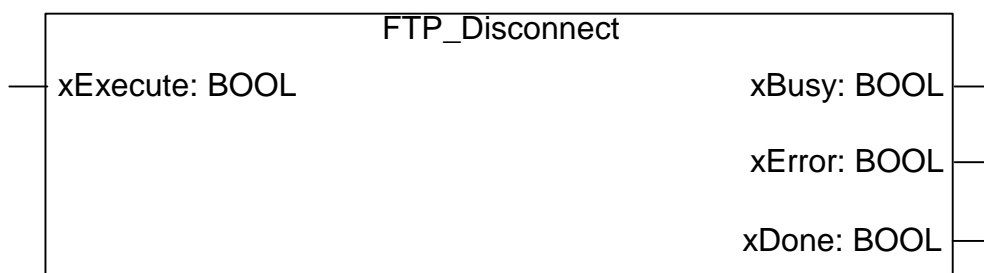
This function is used to transmit passwords to the server. A password is needed whenever the server replies to function FTP_User with response code 331 ("username ok; send password"). If it sends response code 230, no password is required ("login ok").



Var_Input	
xExecute	Function is started with a positive edge
sPassword	Password
Var_Output	
xBusy	Indicates that the function is busy
xError	Indicates an error
xDone	Indicates successful execution of function

2.2.4 FTP_Disconnect

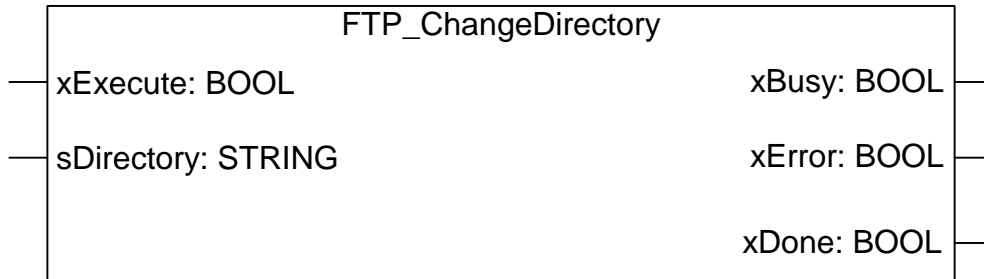
When this function is called, the existing FTP connection is terminated.



Var_Input	
xExecute	Function is started with a positive edge
Var_Output	
xBusy	Indicates that the function is busy
xError	Indicates an error
xDone	Indicates successful execution of function

2.2.5 FTP_ChangeDirectory

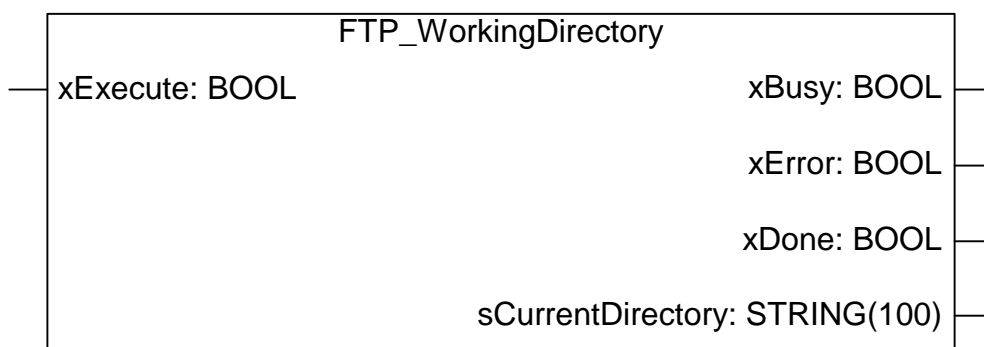
This function is used to change the working directory on the remote host.



<i>Var_Input</i>	
xExecute	Function is started with a positive edge
sDirectory	Directory name. Up to 80 characters
<i>Var_Output</i>	
xBusy	Indicates that the function is busy
xError	Indicates an error
xDone	Indicates successful execution of function

2.2.6 FTP_WorkingDirectory

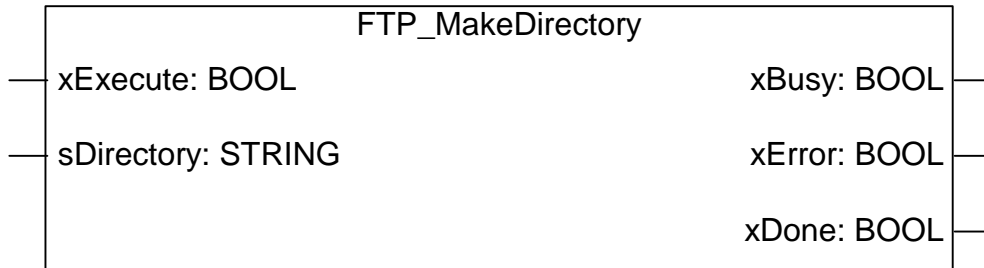
Returns the current FTP server's current working directory.



<i>Var_Input</i>	
xExecute	Function is started with a positive edge
<i>Var_Output</i>	
xBusy	Indicates that the function is busy
xError	Indicates an error
xDone	Indicates successful execution of function
sCurrentDirectory	Returns the current working directory

2.2.7 *FTP_MakeDirectory*

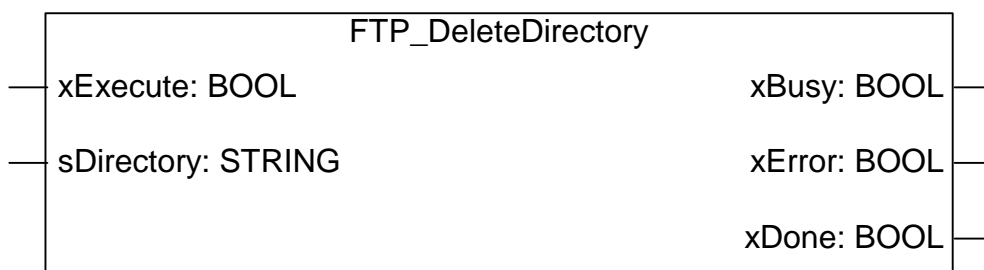
FTP_MakeDirectory is used to create a new directory.



<i>Var_Input</i>	
xExecute	Function is started with a positive edge
sDirectory	Name of directory to be created
<i>Var_Output</i>	
xBusy	Indicates that the function is busy
xError	Indicates an error
xDone	Indicates successful execution of function

2.2.8 *FTP_DeleteDirectory*

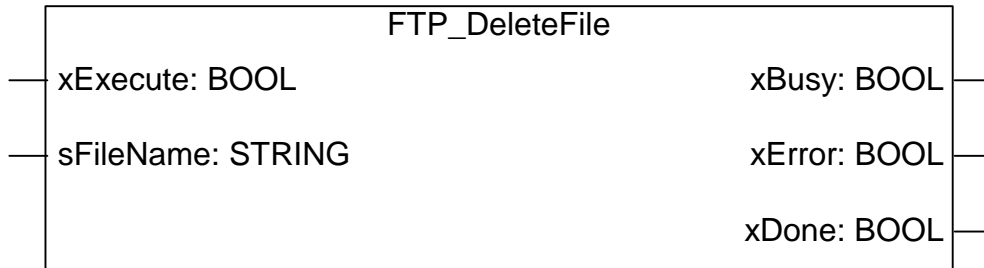
FTP_DeleteDirectory is used to delete directories.



<i>Var_Input</i>	
xExecute	Function is started with a positive edge
sDirectory	Name of directory to be deleted
<i>Var_Output</i>	
xBusy	Indicates that the function is busy
xError	Indicates an error
xDone	Indicates successful execution of function

2.2.9 FTP_DeleteFile

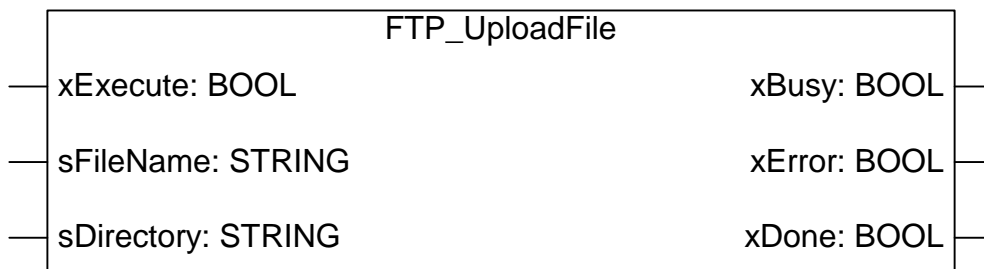
FTP_Delete is used to delete files on the server.



Var_Input	
xExecute	Function is started with a positive edge
sFileName	Name of file on the server to be deleted
Var_Output	
xBusy	Indicates that the function is busy
xError	Indicates an error
xDone	Indicates successful execution of function

2.2.10 FTP_UploadFile

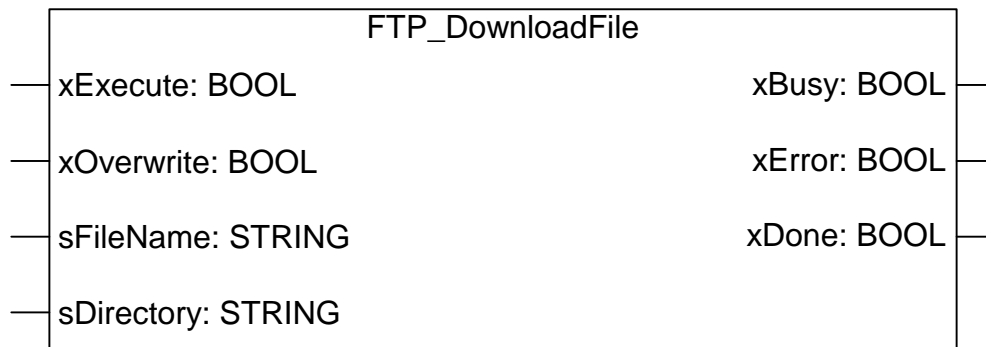
This function is used to upload a file to the server.



Var_Input	
xExecute	Function is started with a positive edge
sFileName	Name of file to be uploaded to the server
sDirectory	Directory of file to be uploaded to the server (must end with a backslash ("\"))
Var_Output	
xBusy	Indicates that the function is busy
xError	Indicates an error
xDone	Indicates successful execution of function

2.2.11 FTP_DownloadFile

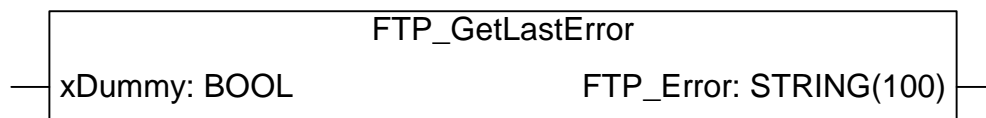
This function is used to download a file from the FTP server.



Var_Input	
xExecute	Function is started with a positive edge
xOverwrite	If TRUE, any existing file is overwritten
sFileName	Name of file to be downloaded from the server
sDirectory	Directory in which the file downloaded from the server is to be saved. Must end with a backslash ("\\")!
Var_Output	
xBusy	Indicates that the function is busy
xError	Indicates an error
xDone	Indicates successful execution of function

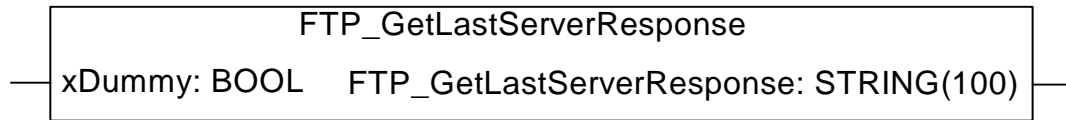
2.2.12 FTP_GetLastError

This function is used to get an error string in plain text. No input variable is required. Because the compiler expects at least one input variable, this function can be given a dummy variable.



2.2.13 *FTP_GetLastServerResponse*

This function provides the meaning of code `FTP_Client_uiLastServerResponse` of the last server response in plain text. No input variable is required. Because the compiler expects at least one input variable, this function can be given a dummy variable.



2.2.14 *Global parameters*

<code>FTP_Client_xActive:</code>	This variable is TRUE when an active connection to an FTP server exists. No further connections can then be established, since the FTP client is laid out for a single instance.
<code>FTP_Client_wErrorID:</code>	This variable holds the current error code. It is used as parameter of function <code>FTP_Error</code> to receive a plain-text description of the error.
<code>FTP_Client_xInit:</code> callback	This variable is set TRUE after successful registration of the function.
<code>FTP_Client_uiLastServerResponse:</code>	In this variable, the server's most recent response code is placed. It is used as a parameter of function <code>FTP_ServerResponse</code> to receive the meaning of the server response in plain text.

The remaining global variables have no significance for using the FTP library.

3 Commissioning

3.1 Inclusion of FTP interfacing under the X series PLCs

Use the library manager to add library FTP_Client.lib. The in additionally required libraries Standard.lib, SysLibCallback.lib, SysLibFile.lib and SysLibSockets.lib must be included too, if they aren't already included.

Note: The FTP client is laid out for a single instance. Do **not**, therefore, call instances from this function block.

Note: The user isn't allowed to call the functions in the Internal Functions folder.

Note: The FTP functions should be run in a separate task. Make sure the watchdog is disabled! You can optimize task settings for your application requirements. If, for example, measured value files are to be transferred at long intervals, you can run the FTP task event-controlled. Otherwise a task interval of about 100 ms or higher is usually suitable. A low priority should be defined.

3.2 Using the FTP library

3.2.1 Logging on to the server

The connection to an FTP server is established with function FTP_Connect. The input parameter is the FTP server's IP address in standard dotted format (example: 192.168.119.300).

The FTP protocol requires a server login before communications can take place. Once the connection is made, a user name must therefore be transmitted with function FTP_UserName. For public servers, the name "anonymous" has become the de facto standard. When the server has accepted the user name, the server's response in global variable *FTP_Client_uiLastServerResonse* must be read. If the login was successful this variable contains code 230 ("login ok"). If the server requires a password, it sends code 331 ("username ok; send password"). To send the password, use function FTP_Password. If the user name is "anonymous", the password can be any string. Once the server has accepted the password, the login is complete.

3.2.2 Directory functions

Function FTP_ChangeDirectory is used to change the working directory on the remote host. The input parameter is the name of the directory to be changed into. This function works in the same way as the MS-DOS change directory (CD) function.

Function FTP_MakeDirectory is used to create a new directory. Observe the length of the directory name: some devices (for example HPG 300) allow only 8 characters. You can use both absolute and relative names.

Function FTP_DeleteDirectory is used to delete directories. You can use both absolute and relative paths.

Function FTP_WorkingDirectory returns the current working directory on the FTP server.

Note: The output variable retains its current value until the next function call, even when the directory is changed with function `FTP_ChangeDirectory`.

3.2.3 File functions

Function `FTP_UploadFile` uploads files to the server. Its input parameters are the filename and the file's directory. Input variable `sDirectory` must end with a backslash, for example `disk_mmc\data\` or `\` for the root directory. This function's busy time depends on the file size and the set task interval.

Function `FTP_DownloadFile` is used for file downloads. The input parameters are the filename on the server and directory to which the file is to be written. For input variable `xDirectory` the same rules apply as for `FTP_UploadFile`. Input variable `xOverwrite` provides write protection: when it is `FALSE`, any existing file with the same name can not be overwritten and the function returns an error. To allow overwriting, this variable must be set `TRUE`.

Depending on the file size, this function also requires several cycles.

Note: If no directory is specified with function `FTP_DownloadFile`, the file is written to the current directory on the PLC. Because local directory functions are not currently supported, this can be any directory. It is therefore important always to specify the target directory.

Function `FTP_Delete` is used to delete files on the server. This action is not reversible. Input variable `sFileName` contains the filename, which can be up to 30 characters long including the file extension. The action is started with a positive edge at `xExecute`. If successful, output variable `xDone` is set `TRUE` for one cycle. If an error occurs, `xError` is set `TRUE`.

3.2.4 Logging off FTP the server

The logoff follows a defined routine using function `FTP_Disconnect`. The FTP session should be closed after completion of the file operation. Some FTP servers terminate the connection when the FTP client has been inactive for some time.

3.2.5 Function `FTP_GetLastError`

To determine the cause of an error, use this function with global variable `FTP_Client_wErrorID`, which returns a string with the error cause in plain text.

Example: `sErrorString:=FTP_GetLastError(xDummy:=TRUE);`

Error codes and associated strings:

0	: 'No Error';
10	: 'unable to retrieve host-address';
11	: 'unable to get local host-name';
12	: 'socket creation failed';
13	: 'receiving server response timed out';
14	: 'transmitting PORT command failed';
15	: 'unexpected server response';
16	: 'server is shutting down';
17	: 'transmitting CWD command timed out';
18	: 'transmitting CWD command failed';
19	: 'transmitting QUIT command failed';
20	: 'unexpected server response. connection closed';
21	: 'creating file failed';
22	: 'FileName already exists! overwrite isn't allowed';
23	: 'listen for data connection failed';
24	: 'port function failed';
25	: 'transmitting RETR command failed';
26	: 'opening file failed';
27	: 'accepting data connection failed';
28	: 'writing data to file failed, disk full';
29	: 'FTP command connection already open';
30	: 'registering callback-functions failed';
31	: 'connecting to remote host failed';
32	: 'transmitting PASS command timed out';
33	: 'transmitting PASS command failed';
34	: 'opening file failed';
35	: 'transmitting USER command timed out';
36	: 'transmitting USER command failed';
37	: 'transmitting XPWD command failed';
38	: 'server closed connection; command socket closed too; FTP-Client not active';
39	: 'transmitting QUIT command timed out';
40	: 'transmitting file failed';
41	: 'transmitting PORT command timed out';
42	: 'receiving server response timed out; connection closed';
43	: 'transmitting RETR command timed out';
44	: 'waiting for incoming connection request timed out';
45	: 'transmitting STOR command timed out';
46	: 'transmitting STOR command failed';
47	: 'transmitting XPWD command timed out';
48	: 'transmitting DELE command timed out';
49	: 'transmitting DELE command failed';
50	: 'transmitting TYPE command timed out';
51	: 'transmitting TYPE command failed';
52	: 'transmitting RMD command timed out';
53	: 'transmitting RMD command failed';
54	: 'transmitting MKD command timed out';
55	: 'transmitting MKD command failed';

All other response codes are acknowledged with "unknown response".

3.2.6 Function *FTP_GetLastServerResponse*

This function provides the meaning of code `FTP_Client_uiLastServerResponse` of the last server response in plain text. No input variable is required.

Example: `sResponse:=FTP_GetLastServerResponse(xDummy:=TRUE);`

Response codes and associated strings:

0:	'no server response already received';
120:	'server not ready yet. try again sometime later';
150:	'file status ok; about to open data connection';
200:	'command ok';
202:	'command not implemented';
220:	'server ready; send username';
221:	'server is closing control connection';
226:	'server is closing data connection';
230:	'login ok';
250:	'requested file action ok';
257:	'requested directory action ok';
331:	'username ok; send password';
332:	'need account for login';
421:	' !!! server is shutting down !!! ';
425:	'can't open data connection';
426:	'connection closed; transfer aborted';
500:	'unknown command';
501:	'error in parameter';
502:	'command not supplied';
503:	'wrong command sequence';
504:	'parameter not allowed';
530:	'user not registered';
532:	'user account needed for storing files on server';
550:	'command not executed; unknown file';
551:	'command not executed';
552:	'command not executed; not enough memory';
553:	'command not executed; invalid filename';
554:	'command not executed; invalid file-mark';
555:	'command not executed; invalid TYPE or STRU command';

All other response codes are acknowledged with 'unknown response'.

4 Version Index

Release; Implementation	Change	Editor
01; 00	First compilation	Matthias Blümling